OpenNMS Install Guide

Getting from Point "A" to Point "Woot"!

Preface iv
1. Overview 1
1.1. About OpenNMS 1
1.2. How to Use This Document 1
1.3. Minimum Requirements 1
2. Preparing for install 3
2.1. Before You Begin 3
2.1.1. Configure RPM-based Distributions with Yum 3
2.1.2. Configure RPM-based Distributions with URPMI (Mandriva) 4
2.1.3. Configure Debian-Based Distributions 4
2.2. Prerequisite Package: Java 4
2.2.1. Installing Java on RPM-based Distributions Using Yum 5
2.2.2. Installing Java on RPM-based Distributions Using URPMI 5
2.2.3. Installing Java on Debian or Ubuntu 5
2.2.4. Installing Java on Other Platforms 5
2.3. Prerequisite Package: PostgreSQL 5
2.3.1. Installing PostgreSQL on RPM-Based Distributions Using Yum 5
2.3.2. Installing PostgreSQL on RPM-Based Distributions Using URPMI 6
2.3.3. Installing PostgreSQL on Debian-Based Distributions 6
2.3.4. Installing PostgreSQL on Windows 6
2.3.5. Configure PostgreSQL 6
2.4. Prerequisite Package: JICMP 8
2.4.1. Installing JICMP on RPM-Based Distributions Using Yum 8
2.4.2. Installing JICMP on RPM-Based Distributions Using URPMI 8
2.4.3. Installing JICMP on RPM-Based Distributions from Source 8
2.4.4. Installing JICMP on Debian-Based Distributions 8
2.4.5. Installing JICMP from Source 8
3. Installing OpenNMS 9
3.1. Where to Find OpenNMS Data 9
3.2. Performing a Fresh Install 9
3.2.1. Installing on RPM-Based Distributions 10
3.2.2. Installing on Solaris 10
3.2.3. Installing on Mac OS X 11
3.2.4. Installing on Windows 11
3.3. Upgrading an Existing Installation 11
3.3.1. Upgrading: Preparation 11
3.3.2. Upgrading RPM-Based Distributions Using Yum 12
3.3.3. Upgrading RPM-Based Distributions Using URPMI 12
3.3.4. Upgrading RPM-Based Distributions Manually 12
3.3.5. Upgrading Debian-Based Distributions 12
3.3.6. Upgrading Windows 12
3.4. Configure Java for OpenNMS 13
3.4.1. Search for a JRE (suggested) 13
3.4.2. Configure a specific JRE 13
3.5. Run the OpenNMS Installer Application 13
4. Getting Started with OpenNMS 14
4.1. Configuring Discovery 14
4.2. Login to the Web Application 14
4.3. Configure OpenNMS to Start Automatically at Boot Time 14

4.3.1. Configuring Automatic Startup on RPM-based Linux Distributions 14
4.3.2. Configuring Automatic Startup on Solaris 14
5. Performance Tuning 15
5.1. Performance "Do"s 15
5.1.1. Lots of RAM 15
5.1.2. Battery-Backed Write Cache 15
5.1.3. Multiple Spindles 15
5.1.4. Use `noatime` on OpenNMS Data Spindles on Linux and Solaris 16
5.1.5. RAID Drives 16
5.1.6. PostgreSQL Performance Tuning 16
5.2. Performance "Don't"s 17
6. Building From Source 18
6.1. Are you sure you want to do this? 18
7. Troubleshooting an OpenNMS Installation 19
7.1. Common Installation Issues 19
7.1.1. Dependency Problems 19
7.1.2. Error: "Started OpenNMS, but it has not finished starting up" 19
7.1.3. DHCP Poller Won't Start 20
7.1.4. Error: "runjava: Could not find an appropriate JRE" 20
7.1.5. Error: "The database server's error messages are not in English ..." 20
7.1.6. Error: "Column X in new table has NOT NULL constraint ..." 20
7.1.7. Error: "One or more backup tables from a previous install still exists" 21
7.1.8. Error: "Table X contains N rows (out of M) that violate new constraint Y"..... 21
7.1.9. Error: "- adding iplike database function... <snip>
org.postgresql.util.PSQLException: ERROR: could not access file '<snip>/lib/
iplike.so': Permission denied" 22
7.1.10. Error: "- adding iplike database function... <snip>
org.postgresql.util.PSQLException: ERROR: could not load library ..." 22
7.1.11. Error: "Exception in thread "main" org.postgresql.util.PSQLException:
ERROR: relation "pg_user" does not exist" when running installer. 22
7.1.12. Error: java.io.FileNotFoundException: ... (Permission denied) 23
7.2. Where to Get Help 23
7.2.1. The Release Notes 23
7.2.2. The OpenNMS Web Site 23
7.2.3. The OpenNMS Mailing Lists 23
7.2.4. Commercial Support 24

Preface

OpenNMS is the creation of numerous people and organizations, operating under the umbrella of the OpenNMS project. The original code base was developed and published under the GPL by the Oculan Corporation until 2002, when the project administration was passed on to Tarus Balog.

The current corporate sponsor of OpenNMS is The OpenNMS Group, which also owns the OpenNMS trademark.

OpenNMS is a derivative work, containing both original code, included code and modified code that was published under the GNU General Public License. Please see the source for detailed copyright notices, but some notable copyright owners are listed below:

- Copyright © 2002-2009 The OpenNMS Group, Inc.
- Original code base for OpenNMS version 1.0.0 Copyright © 1999-2001 Oculan Corporation.
- Mapping code Copyright © 2003 Networked Knowledge Systems, Inc.
- ScriptD code Copyright © 2003 Tavve Software Company.

Please send any omissions or corrections to this document to Tarus Balog.

Chapter 1. Overview

1.1. About OpenNMS

OpenNMS is the world's first enterprise-grade network management system developed under the open source model. As with any complex and powerful system, getting it installed and configured can take a little effort. It is the purpose of this document to explain what is required to get OpenNMS installed.

1.2. How to Use This Document

So, how should you use this document? It is arranged in the following sections:

- This overview
- The programs on which OpenNMS depends, and how they need to be modified
- Installation and upgrade instructions, including details for specific operating systems and distributions
- Getting started with OpenNMS, including initial configuration and logging into the web interface
- Building OpenNMS from source
- Troubleshooting and Where to Get Help

This installation guide relies strongly on the idea of "packages." Most modern operating systems and distributions have a system where software can be installed and managed through the use of packages that group the files belonging to a given application together (as well as managing changes to those files, removal, upgrades, etc.).

Please see the latest release notes to see if your operating system is supported. Currently, OpenNMS runs on many Linux distributions, Solaris, Mac OS X and Windows.

This guide assumes that if you use packages, you do so consistently. This is because OpenNMS will attempt to determine if the software it requires is installed by using the operating system's built in package management system. If you've installed, say, Java, but not via packages, OpenNMS will be unable to determine that Java is installed and it will fail.

To get back to the original question of "how should you use this document," first go through the second section to insure that you have all of the prerequisite applications properly installed and configured. Use the third section to help get those packages installed for your particular operating system, as well as the OpenNMS software. Finally, use the last section to help correct any errors your might encounter.

1.3. Minimum Requirements

While it is impossible to exactly size OpenNMS for a particular environment, the following represents the minimum requirements for installation, assuming a network of about 200 devices. Note that OpenNMS can monitor more than 100 times that given the proper hardware.

Processor	A 1 GHz Pentium III (or equivalent processor) or better. OpenNMS can also take advantage of multiple processors.

Memory	A minimum of 256 MB of RAM, although 512 MB is strongly recommended. The OpenNMS Java Virtual Machine benefits from large amounts of memory, up to 2 GB, and more if using a 64-bit processor. Given a budget choice between more RAM and a faster CPU, choose more RAM.
Disk Space	OpenNMS requires about 200 MB of disk space for the program files. In addition, each data variable collected requires, by default, a little under 300 KB of disk space. It is safe to assume that each interface being managed will require around 2 MB of disk space, so for 200 interfaces you are looking at 400 MB (conservatively). Depending on the number of events stored, you can assume 100 MB to 200 MB are required for the database. Finally, the OpenNMS logs can grow quite large, especially in debug mode. Edit the `log4j.properties` file in the OpenNMS configuration directory (usually `/opt/opennms/etc` or `/etc/opennms`) to change those settings. By default, the Log4J file rotation is configured to use 100MB per log file, which ends up using a little under 2 GB. **Note**: Due to the write-heavy nature of time-series data and the database, it is recommended that you do not use RAID-5 with OpenNMS. RAID-1 or RAID-1+0 is recommended if using RAID. In addition, LVM adds a small but appreciable amount of overhead and it is recommended that you do not use it.

Chapter 2. Preparing for install

2.1. Before You Begin

2.1.1. Configure RPM-based Distributions with Yum

Before you begin installing, you will want to set up Yum to install from the OpenNMS repositories. This covers most RPM-based distributions, including Red Hat Enterprise Linux, Fedora, and CentOS.

2.1.1.1. Preparation: Yum Fastest Mirror Plugin

Before you start, you may want to install the yum-fastestmirror RPM if your distro supports it. This can often speed up downloads of large packages. See the CentOS Wiki for more details. This step is not strictly necessary, but can make your overall yum experience better.

```
[user@localhost]$ sudo yum install yum-fastestmirror
 Setting up Install Process
 ...
 Running Transaction
   Installing: yum-fastestmirror           ######################### [1/1]

 Installed: yum-fastestmirror.noarch 0:1.1.9-2.fc8
 Complete!
```

2.1.1.2. Preparation: Determine Which Release to Install

There are 4 types of releases available through yum.

- *stable:* the latest officially released stable version of OpenNMS
- *unstable:* the latest officially released development version of OpenNMS
- *testing:* a nightly build of the code that will be part of the next stable version of OpenNMS
- *snapshot:* a nightly build of the very latest development version of OpenNMS

2.1.1.3. Install the OpenNMS Repository RPM

To simplify installation through Yum, we've created an RPM that contains the configuration necessary for Yum to be able to find the other OpenNMS packages. Based on the release you chose in the section above, choose the approprate RPM from the OpenNMS Yum Repository.

For example, to install the latest snapshot release on Fedora 7, you would run:

```
rpm -Uvh http://yum.opennms.org/repofiles/opennms-repo-snapshot-fc7.noarch.rpm
```

Or, to install the latest unstable release on CentOS or RHEL 5, you would run:

```
rpm -Uvh http://yum.opennms.org/repofiles/opennms-repo-unstable-rhel5.noarch.rpm
```

Now you should see OpenNMS packages available when you get a list of yum packages:

```
 [user@localhost]$ sudo yum list opennms
 ...
 Available Packages
 opennms.noarch                          1.5.96-1             opennms-unstable
```

Note

If you are using older yum-based distributions (like CentOS 3, for example), you may need to append the yum configuration to `/etc/yum.conf`. Older versions of yum don't recognize `/etc/yum.repos.d/` as a valid location for yum configuration. You can fix this by using `cat` to append the repository configurations to `/etc/yum.conf`:

```
[root@localhost]# cat /etc/yum.repos.d/* >> /etc/yum.conf
```

2.1.2. Configure RPM-based Distributions with URPMI (Mandriva)

2.1.2.1. Enable the Primary Mandriva Repositories

First, you'll want to enable the primary Mandriva URPMI repositories. The easiest way to do so is to follow the instructions at EasyURPMI. For example, on Mandriva Linux 2007, you would end up running something like this:

```
urpmi.addmedia main ftp://mirrors.usc.edu/pub/linux/distributions/mandrakelinux/official/2007.1/i586/media/main/release with media_info/hdlist.cz
 urpmi.addmedia --update main_updates ftp://mirrors.usc.edu/pub/linux/distributions/mandrakelinux/official/2007.1/i586/media/main/updates with media_info/hdlist.cz
```

2.1.2.2. Enable the OpenNMS Mandriva Repositories

Now, you'll need to enable the OpenNMS Mandriva repositories. First, add the OpenNMS stable repository (replace mandriva2007 with your release):

```
urpmi.addmedia --probe-hdlist opennms-stable http://yum.opennms.org/stable/mandriva2007
```

If you want OpenNMS stable snapshots, add the testing repository next (replace mandriva2007 with your release):

```
urpmi.addmedia --probe-hdlist opennms-testing http://yum.opennms.org/testing/mandriva2007
```

If you want the latest unstable version, add the unstable as well (replace mandriva2007 with your release):

```
urpmi.addmedia --probe-hdlist opennms-unstable http://yum.opennms.org/unstable/mandriva2007
```

And if you want to install nightly snapshots, then add the snapshot one (replace mandriva2007 with your release):

```
urpmi.addmedia --probe-hdlist opennms-snapshot http://yum.opennms.org/snapshot/mandriva2007
```

2.1.3. Configure Debian-Based Distributions

2.1.3.1. Add the OpenNMS Repository to Your `sources.list`

First, you need to tell apt-get how to find OpenNMS. Add the following contents to your `/etc/apt/sources.lists` file:

```
deb http://debian.opennms.org stable main
deb-src http://debian.opennms.org stable main
```

If you wish to use the latest development version of OpenNMS, add unstable instead:

```
deb http://debian.opennms.org unstable main
deb-src http://debian.opennms.org unstable main
```

2.1.3.2. Add the OpenNMS PGP Key to APT

The OpenNMS Debian repository is signed with a PGP key (fingerprint `22EE DDA6 8698 B02F B2EC 50B7 062B 8A68 4C4C BBD9`). You will need to tell APT about the key:

```
wget -O - http://debian.opennms.org/OPENNMS-GPG-KEY | sudo apt-key add -
```

2.2. Prerequisite Package: Java

OpenNMS is written mainly in Java, although there are a few JNI calls to some C code in order to implement things such as ICMP. and so it follows that Java would need to be installed.

OpenNMS requires Java SE 5.0 or higher (JDK 1.5). It is recommended that the JDK from Sun is used with OpenNMS. If OpenNMS is to be run on a 64-bit system, be sure to install the 64-bit JDK.

2.2.1. Installing Java on RPM-based Distributions Using Yum

The Sun JDK is available in our Yum repository. If you have configured Yum as specified above, you just need to run:

```
yum install jdk
```

Because of a bug in the 64-bit RPM signing, if you are on x86_64, you will need to disable GPG checking. You can do so with the `--nogpgcheck` option to yum:

```
yum --nogpgcheck install jdk
```

2.2.2. Installing Java on RPM-based Distributions Using URPMI

The Sun JDK is available in our URPMI repository. If you have configured URPMI as specified above, you just need to run:

```
urpmi --auto jdk
```

2.2.3. Installing Java on Debian or Ubuntu

Sun's Java can now be installed using "apt" on Debian Etch or higher.

```
apt-get install sun-java5-jdk
```

This should also work on Ubuntu 6.10 (Edgy Eft) or higher. Alternatively, you could install sun-java6-jdk, which has performance improvements over the java5 version.

2.2.4. Installing Java on Other Platforms

Note

It is important to install the JDK instead of the JRE, as the web UI will need to compile JSPs into Java code.

You will need to use Sun's Java SE, version 5 (1.5) or later. You can download it from Sun's Java website. Step through the licensing process and then download the proper version of Java for your operating system.

2.3. Prerequisite Package: PostgreSQL

PostgreSQL (or "Postgres") is a relational database that OpenNMS uses to store information about devices on the network, as well as information about events, notifications and outages.

When installing OpenNMS, two things must happen. First, OpenNMS has to be able to contact the database over TCP/IP (even on localhost) and second, the installation process must be able to create the database.

OpenNMS requires version 7.4 or later of PostgreSQL, although 8.1 or higher is recommended for performance reasons.

2.3.1. Installing PostgreSQL on RPM-Based Distributions Using Yum

On modern versions of Red Hat Enterprise Linux, CentOS, and Fedora, you should just need to install the `postgresql-server` RPM:

```
[user@localhost]$ sudo yum -y install postgresql-server
```

```
Setting up Install Process
...
Running Transaction
  Installing: postgresql-server           ########################## [1/1]

Installed: postgresql-server.x86_64 0:8.2.5-1.fc8
Complete!
```

Note

Red Hat Enterprise Linux 3 and CentOS 3 call their PostgreSQL packages "rhdb" for the "Red Hat DataBase" so if you are on one of these older distributions, you will have to substitute "rhdb" for "postgresql" when installing:

```
sudo yum -y install rhdb-server
```

2.3.2. Installing PostgreSQL on RPM-Based Distributions Using URPMI

On Mandriva, you use URPMI to install the PostgreSQL server:

```
sudo urpmi --auto postgresql-server
```

2.3.3. Installing PostgreSQL on Debian-Based Distributions

On Debian or Ubuntu, use apt to install the PostgreSQL server:

```
sudo apt-get update
sudo apt-get install postgresql-8.1
```

2.3.4. Installing PostgreSQL on Windows

On Windows, all you should need to do is get the latest Windows installer from PostgreSQL.org.

Note

If you are running on a FAT32 filesystem, see the detailed installation instructions on the wiki.

First, unpack the installer. The installer does not run properly from inside a zipped folder, so you will need to extract the ZIP file. You should be able to just copy the postgresql-X.X.msi and postgresql-X.X-int.msi files to your desktop and run them from there.

Then, run the postgresql-X.X.msi and follow the instructions. For the most part, the defaults should be just fine, although if you're allowing the installer to initialize your database, make sure the encoding is set to "UTF-8".

2.3.5. Configure PostgreSQL

Once you have installed PostgreSQL, you will need to make two changes to Postgres configuration files: `postgresql.conf` and `pg_hba.conf`.

These files are only created once PostgreSQL has been started, so if your installation method for Postgres did not start the database, do so before continuing. Usually, startup scripts will be placed in `/etc/init.d`.

Locate the Postgres "data" directory. Often this is `/var/lib/pgsql/data`. You should then find the two files we need to modify in that directory.

2.3.5.1. The `postgresql.conf` File

This file controls some basic parameters of PostgreSQL. We need to change three of these parameters.

1. First we need to make sure PostgreSQL is listening on an IP socket, and not just a local unix socket.

   For PostgreSQL 7.4 and 8.0, make sure the following line is set and uncommented:

   ```
   tcpip_socket = true
   ```

   On PostgreSQL 8.1 and up, use this instead:

   ```
   listen_addresses = 'localhost'
   ```

2. Next, find the line in the file that contains `max_connections`. It needs to be at least:

   ```
   max_connections = 256
   ```

3. Find the line that contains `shared_buffers`. It needs to be at least:

   ```
   shared_buffers = 1024
   ```

2.3.5.2. Customizing the `pg_hba.conf` File

The `pg_hba.conf` file controls which machines and users can access the database on a given machine via TCP/IP.

Since that is how OpenNMS accesses the database (via `localhost`) it is necessary to modify this file to allow OpenNMS to work. The easiest thing to do is to just allow anyone from the localhost to access the database (do not add the last line if your system does not support IPv6):

```
# TYPE DATABASE USER IP-ADDRESS IP-MASK METHOD
local all all trust
host all all 127.0.0.1 255.255.255.255 trust
host all all ::1 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff trust
```

Make sure that no other lines are uncommented in this file.

You will need to stop and restart Postgres after making these changes.

2.3.5.3. Creating the PostgreSQL Database

Most distributions will automatically initialize the default database on first startup, but if yours doesn't (for example, on Solaris), you will need to do so manually.

As the `postgres` user, go to the `/usr/local/pgsql/bin` directory (or wherever your PostgreSQL binaries are installed), and run:

```
./initdb -D /usr/local/pgsql/data -E "UNICODE"
```

Then you'll need to start the database:

```
./pg_ctl -D /usr/local/pgsql/data start
```

2.3.5.4. Adding the iplike function

OpenNMS makes heavy use of a stored procedure called "iplike". Since it is written in C, it has been removed from the main OpenNMS code and placed in its own project.

If a C-based iplike is not installed, the OpenNMS installer will add a version written in the PostgreSQL command language. It will work, but not as quickly as the compiled iplike will.

To install iplike, simply download the proper package for your distribution. There should be a package for PostgreSQL versions 7.4-8.1, and one for 8.2+. In addition, there will be separate 32-bit and 64-bit versions. It is also possible to download a tarball from the OpenNMS SourceForge project page,

and do the usual "./configure", "make", and "make install". Once installed it should not be required to update it on every OpenNMS upgrade.

2.4. Prerequisite Package: JICMP

Java has never had a really good API for ICMP. Since ICMP is the basis for the "ping" command, it is rather imperative that any Java-based network management platform address the need for ICMP. OpenNMS does this by using some code written in C, and accessing it using the Java Native Interface (JNI).

As of OpenNMS 1.3.6, the ICMP code has been moved to it's own library outside of OpenNMS. This makes the main OpenNMS application pure Java, and as such it only has to be built once, instead of for each platform.

Packages for JICMP are available for most distributions. If your distribution does not have packages available, you can download the source from the SourceForge download page for JICMP.

2.4.1. Installing JICMP on RPM-Based Distributions Using Yum

On most RPM-Based Distributions, all you should need to run is:

```
yum install jicmp
```

2.4.2. Installing JICMP on RPM-Based Distributions Using URPMI

On Mandriva, you can install JICMP with the command:

```
urpmi --auto jicmp
```

2.4.3. Installing JICMP on RPM-Based Distributions from Source

If JICMP has not already been compiled on your RPM-based platform, you can build a native RPM from the source tarball like so:

```
rpmbuild -tb jicmp-X.X.X.tar.gz
```

If you are on a 64-bit platform, you can build a 64-bit RPM instead like so:

```
rpmbuild --target=x86_64 jicmp-X.X.X.tar.gz
```

2.4.4. Installing JICMP on Debian-Based Distributions

On Debian or Ubuntu, you can install JICMP through apt:

```
sudo apt-get install libicmp-jni
```

2.4.5. Installing JICMP from Source

To build from source, download the latest source tarball from SourceForge, unpack it, and run the usual:

```
./configure
make
make install
```

Chapter 3. Installing OpenNMS

Note

You need to be root when running most of the commands in this chapter.

3.1. Where to Find OpenNMS Data

OpenNMS stores data in a number of locations:

`$OPENNMS_HOME/etc/`	OpenNMS configuration files. On some distributions, upstream changes in files in this directory are detected and means are provided for migrating those changes to the new release. On RPM-based distributions, if an OpenNMS configuration file has changed, RPM will create a "`.rpmnew`" file which contains the version of that configuration file that shipped with the new version of OpenNMS. You will need to look at the changes between your file and the new one and merge them manually, at the moment. The command "`diff -u <old file> <new file> | less`" can assist you in seeing what has changed. On Debian-based distributions, dpkg will automatically prompt you when a configuration file has changed upstream between versions, and will offer you a set of options to deal with it.
`$OPENNMS_HOME/share/rrd/`	RRD data files that store response time data and performance data collected from SNMP (and elsewhere). The installer should not touch the RRD files in `$OPENNMS_HOME/share/rrd`. Unless you are migrating from RRDTool to jRobin, you should not have to worry about them.
`$OPENNMS_HOME/webapps/opennms/` and `$OPENNMS_HOME/jetty-webapps/opennms/`	The OpenNMS web application. While data is not stored here, some users may customize the web interface and these customizations should be saved before upgrading OpenNMS.
`$PGDATA/`	Data about nodes, services, events, notifications, etc., are stored in the `opennms` table in PostgreSQL.

3.2. Performing a Fresh Install

Follow the instructions in this section appropriate for your operating system if you are performing a fresh install. If you are upgrading an existing installation of OpenNMS, see the next section.

3.2.1. Installing on RPM-Based Distributions

3.2.1.1. Determine What to Install

As of version 1.6, OpenNMS is packaged in a modular fashion. The following packages are available for installation:

- *opennms-core*: The core OpenNMS code, responsible for network discovery, polling, data collection, notification, and more.
- *opennms-docs*: Documentation.
- *opennms-webapp-jetty*: The OpenNMS web UI, designed to be started by the opennms-core engine.
- *opennms-webapp-standalone*: The OpenNMS web UI, designed to be run in Tomcat or another suitable servlet container.
- *opennms*: A convenience package which installs everything you need for a functional OpenNMS installation on a single system.
- *opennms-remote-poller*: The standalone remote poller, which can report back to an OpenNMS instance.
- *opennms-plugin-ticketer-centric*: The CentricCRM/Concursive ticketer plugin.

3.2.1.2. Installing on RPM-Based Distributions Using Yum

As long as Yum is configured to point at the OpenNMS repository, all you should need to run is:

```
sudo yum install [packages]
```

...where `[packages]` is the list of packages above that you wish to install.

3.2.1.3. Installing on RPM-Based Distributions Using URPMI

If URPMI is configured to point at the OpenNMS repository, you can install by running:

```
sudo urpmi --auto [packages]
```

...where `[packages]` is the list of packages above that you wish to install.

3.2.1.4. Installing on RPM-Based Distributions Manually

Download the packages you wish to install from the SourceForge download page for OpenNMS.

Then, use `rpm -i` to install the packages:

```
rpm -ivh opennms*.rpm
```

Note

As of OpenNMS 1.3.10, you can use the `--relocate` flag to RPM if you wish to put the logs and collection data in an alternate location:

```
rpm -ivh --relocate /var/opennms=/mnt/netapp/opennms-data /var/log/opennms=/mnt/netapp/opennms-logs opennms*.rpm
```

3.2.2. Installing on Solaris

Download the appropriate package for your Solaris version from the SourceForge download page for OpenNMS.

Then, install the package using `pkgadd`:

```
# cd /usr/local
# gzip -d opennms-*-local.gz
# pkgadd -d `pwd`/opennms-*-local
```

3.2.3. Installing on Mac OS X

OpenNMS is supported on Mac OS X via the Fink project.

Once you've installed Fink, you should be able to install OpenNMS by running:

```
fink install opennms
```

3.2.4. Installing on Windows

OpenNMS is supported on Windows as well, although the lack of true package management makes it a bit more work to maintain.

To install on Windows, download the latest `standalone-opennms-installer-X.X.X.jar` file from the OpenNMS SourceForge download page.

You should be able to then just double-click the jar file in Explorer, and follow the instructions in the install wizard.

Once you're done, you may want to make a pristine copy of the `$OPENNMS_HOME/etc` directory so it's easy to tell what's changed in later releases later.

3.3. Upgrading an Existing Installation

3.3.1. Upgrading: Preparation

There are a number of things you can do that can help ease the transition when doing upgrades.

3.3.1.1. Prune Unneeded Events

Should you not be doing this already, either using vacuumd or cron, prune away any unneeded events. The events table will most probably be the largest, and there's no point backing up data that you don't need.

For example, to delete any events older than 6 weeks that have no associated outages, you can run, from the `psql` command-line:

```
--# this deletes any events that are not associated with outages
DELETE FROM events WHERE NOT EXISTS
  (SELECT svclosteventid FROM outages WHERE svclosteventid = events.eventid
UNION
  SELECT svcregainedeventid FROM outages WHERE svcregainedeventid = events.eventid
UNION
  SELECT eventid FROM notifications WHERE eventid = events.eventid)
AND eventtime < now() - interval '6 weeks';
```

3.3.1.2. Back Up Your Database

Depending on the version you are upgrading *from* and the version you are upgrading *to*, you may run into issues with the database upgrade. It is *strongly recommended* that you back up the database.

```
pg_dump -U postgres -Fc -C -f opennms-database-backup.pg opennms
```

3.3.1.3. Back Up Your OpenNMS `etc` Directory

Back up your `$OPENNMS_HOME/etc` directory before doing any upgrades, so you can go back to your previous version if something goes wrong, and so you have a reference of your previous configuration when adapting config file changes to the new version.

```
tar -C $OPENNMS_HOME -cvzf /tmp/opennms-etc-backup.tar.gz etc
```

3.3.1.4. Assess the Events Table Size

If you have a large number of events, you may need to increase the amount of memory passed to the OpenNMS `install` tool. For example, if you have 250k events, you will need almost 600MB of heap. When the time comes to run the `install` tool, assuming you have enough available memory, it's probably safe to just use a very large value to pass to `install`, like:

```
install -Xms1024m -Xmx2048m -dis
```

3.3.2. Upgrading RPM-Based Distributions Using Yum

If you have the OpenNMS Yum repositories configured, all you should need to do to get the new release is run:

```
sudo yum upgrade opennms
```

If you are comfortable letting your distribution give you OS updates along with OpenNMS, you can just run:

```
sudo yum upgrade
```

3.3.3. Upgrading RPM-Based Distributions Using URPMI

If you have the OpenNMS URPMI repositories configured, you can upgrade by running:

```
sudo urpmi --auto opennms
```

As with Yum, if you are comfortable letting URPMI give OS updates along with OpenNMS, you can run:

```
sudo urpmi --auto --auto-select opennms
```

3.3.4. Upgrading RPM-Based Distributions Manually

You can upgrade RPM systems manually by downloading the RPM packages from SourceForge and upgrade them by running:

```
sudo rpm -Uvh opennms*.rpm
```

3.3.5. Upgrading Debian-Based Distributions

If your Debian or Ubuntu system is configured with the OpenNMS Apt repository, you should just need to run:

```
sudo apt-get update
sudo apt-get -u install opennms
```

3.3.6. Upgrading Windows

Since there is no automated installation/upgrade procedure for Windows, you will have to do some preparation before installing the latest version.

- Back up and then remove `$OPENNMS_HOME/etc`.
- Remove `$OPENNMS_HOME/lib`, `$OPENNMS_HOME/webapps/opennms/WEB-INF/lib`, and `$OPENNMS_HOME/jetty-webapps/opennms/WEB-INF/lib`.

Once you've cleaned up the etc and lib directories, the next step is to download the latest `standalone-opennms-installer-X.X.X.jar` from the OpenNMS SourceForge download page.

Next, run the installer jar, and install over your existing OpenNMS location.

If you backed up the pristine `etc` directory on your previous installation, you can compare it to the current one to see if there are any configuration changes you wish to integrate into your new install.

Copy your backed up `etc` directory back into the `$OPENNMS_HOME/etc` directory.

You should now be able to run your upgraded OpenNMS.

3.4. Configure Java for OpenNMS

Before you can run the post-install, OpenNMS needs to be configured to use an appropriate Java Runtime Environment (JRE). The OpenNMS tool `runjava` is used to set this up, and it can either search for a suitable JRE or you can tell it exactly which JRE to use.

3.4.1. Search for a JRE (suggested)

Execute `runjava` with the "`-s`" option to search for a JRE:

```
# $OPENNMS_HOME/bin/runjava -s
```

3.4.2. Configure a specific JRE

Execute runjava with the "`-S <path to JRE>`" option to specify the exact JRE you want OpenNMS to use:

```
# $OPENNMS_HOME/bin/runjava -S <path to JRE's java executable>
```

3.5. Run the OpenNMS Installer Application

No matter which installation method above you choose, and whether you are performing a fresh install or an upgrade, you still need to run the OpenNMS installer. This tool will setup the `opennms` database within PostgreSQL among other things.

```
# $OPENNMS_HOME/bin/install -l /usr/local/lib -dis
```

The "-l" parameter will look for the jicmp and/or jrrd libraries in the location specified. The "-dis" will initialize and check the database. Note at the end of the output from the installer it will indicate if iplike has been installed properly.

For a full list of options the installer accepts, run "`$OPENNMS_HOME/bin/install -h`".

Chapter 4. Getting Started with OpenNMS

4.1. Configuring Discovery

By default, OpenNMS will not discover hosts until you configure it to do so, or explicitly add them in the Admin UI. You will most likely want to tell OpenNMS where to look to discover hosts on your network.

Edit `$OPENNMS_HOME/etc/discovery-configuration.xml`. You should see an example <include-range> tag with a <begin> and an <end> which is commented out.

You will most likely want to uncomment it and change the beginning and end ranges (within the `<begin>` and `<end>` tags, respectively). Additionally, you can add as many `<include-range>`es as you'd like.

If you would rather list the individual host that you want to have discovered, you can insert `<specific>` tags above the `<include-range>` tag.

Lastly, if you prefer to use the web interface to add individual hosts for OpenNMS to monitor, you leave the contents of this file commented out.

4.2. Login to the Web Application

By default, OpenNMS's built-in web server listens on port 8980, so point your browser at `http://<host>:8980/opennms/` (where <host> is the host you're running OpenNMS on). The initial user name is "`admin`" and the password is "`admin`".

4.3. Configure OpenNMS to Start Automatically at Boot Time

If everything looks good, you can configure OpenNMS to start automatically at boot time. By default on most platforms OpenNMS does not start automatically until you configure it to do so.

4.3.1. Configuring Automatic Startup on RPM-based Linux Distributions

The OpenNMS packages add an init script in `/etc` (usually `/etc/init.d`), however you need to execute `chkconfig` to enable the service to start automatically:

```
# /sbin/chkconfig --add opennms
```

4.3.2. Configuring Automatic Startup on Solaris

```
# ln -s $OPENNMS_HOME/bin/opennms /etc/init.d/opennms
# ln -s ../init.d/opennms /etc/rc3.d/S99opennms
# ln -s ../init.d/opennms /etc/rc3.d/K01opennms
```

Chapter 5. Performance Tuning

5.1. Performance "Do"s

5.1.1. Lots of RAM

OpenNMS is not terribly heavy on CPU usage, but is *extremely* I/O-bound, and will also take advantage of as much RAM as you can give it. OpenNMS itself doesn't use a huge amount of RAM per-node, but allowing the OS to cache filesystem interaction makes a very large performance difference.

5.1.2. Battery-Backed Write Cache

If you are running on hardware RAID, it is strongly recommended that you have a battery-backed write cache. For example, one user reported that on an HP DL380 G4, the I/O wait of the server dropped from 15% to essentially nothing, using a 128MB battery-backed write cache.

5.1.3. Multiple Spindles

You will get the most out of OpenNMS if you spread your I/O out into multiple spindles and/or separate disks/channels.

5.1.3.1. PostgreSQL

PostgreSQL writes primarly to 2 classes of files and directories.

the database	The main PostgreSQL database is in `$PGDATA/base` (`$PGDATA` is usually something like `/var/lib/pgsql/data`).
the journal	PostgreSQL keeps a journal of transactions, in `$PGDATA/pg_xlog`.

If you can separate the pg_xlog directory onto another spindle or mount point, you will increase your PostgreSQL performance considerably. To do so, you should be able to just shut down PostgreSQL, move that directory, symlink it to the old location, and start it back up.

```
sudo /etc/init.d/postgresql stop
sudo mv /var/lib/pgsql/data/pg_xlog /mnt/xlogspindle/pg_xlog
sudo ln -s /mnt/xlogspindle/pg_xlog /var/lib/pgsql/data/pg_xlog
sudo /etc/init.d/postgresql start
```

5.1.3.2. Round-Robin (Collection and Performance) Data

The RRD data is the single heaviest source of I/O in most OpenNMS installations. Making sure that it is on a different spindle from PostgreSQL makes a huge difference.

- RRD data storage causes a large number of small random disk writes, usually a few writes for each update.
- By default, OpenNMS stores each collected variable in its own file, unless the store by group feature is enabled.
- Normally, there will be 2-3 writes for each update: one for the file header, one for the previous RRA, one for the next RRA.
- When multiple samples are consolidated into a single stored data point in the RRA, there will be additional writes. By default, such consolidations happen hourly and daily on the GMT day

boundary. This will cause higher than normal amount of writes after the top of the hour and after the GMT day boundary.

The OpenNMS RRDs live, by default, in `$OPENNMS_HOME/share`. If you are using the RPMs, this will be `/var/opennms` instead.

```
sudo mv /var/opennms /mnt/rrdspindle/opennms
sudo rm -f /opt/opennms/share
sudo ln -s /mnt/rrdspindle/opennms /opt/opennms/share
```

5.1.4. Use `noatime` on OpenNMS Data Spindles on Linux and Solaris

If you are dedicating spindles or drives to OpenNMS, you can mount them with the `noatime` option on Linux or Solaris for an additional performance boost. This will keep the OS from updating the file access time on individual RRD and database files every time they are used.

On Linux, you do so by editing `/etc/fstab` and adding `noatime` to the options section of the filesystem. For example:

```
LABEL=/                         /                         ext3    defaults        1 1
LABEL=/var/opennms              /var/opennms              ext3    defaults,noatime 1 2
LABEL=/var/lib/pgsql            /var/lib/pgsql            ext3    defaults,noatime 1 2
LABEL=/var/lib/pgsql/data/pg_xlog /var/lib/pgsql/data/pg_xlog ext3    defaults,noatime 1 2
```

On Solaris, you edit `/etc/vfstab` and add `noatime` as an option at the end of the mountpoint information, like so:

```
/dev/dsk/c1d0s0 /dev/rdsk/c1d0s0 /                          ufs 1 no
/dev/dsk/c1d1s0 /dev/rdsk/c1d1s0 /opt/opennms/share          ufs 2 no noatime
/dev/dsk/c1d2s0 /dev/rdsk/c1d2s0 /usr/local/pgsql/data       ufs 2 no noatime
/dev/dsk/c1d3s0 /dev/rdsk/c1d3s0 /usr/local/pgsql/data/pg_xlog ufs 2 no noatime
```

5.1.5. RAID Drives

Use a mirrored stripe (RAID-10), with enough disks to handle the amount of data you need to collect. A single disk, a pair of mirrored disks (RAID-1), or a RAID-5 is only appropriate for an installation doing a small amount of data collection.

5.1.6. PostgreSQL Performance Tuning

There are a number of other things you can do to tune PostgreSQL. For a good writeup on PostgreSQL performance tuning, see [this page at revsys.com](revsys.com).

5.1.6.1. PostgreSQL 8.1-specific Recommendations

If you have a reasonable amount of RAM (2GB+), the following settings should give much better performance than the defaults that come with the PostgreSQL configuration:

```
shared_buffers = 20000
work_mem = 16348
maintenance_work_mem = 65536
vacuum_cost_delay = 50
checkpoint_segments = 20
checkpoint_timeout = 900
wal_buffers = 64
stats_start_collector = on
stats_row_level = on
autovacuum = on
```

5.1.6.2. PostgreSQL 8.2+ Recommendations

On systems with 4GB or more of RAM, we've found that changing the max_fsm_pages and max_fsm_releations, as well as work_mem and maintenance_work_mem improves performance dramatically:

```
work_mem = 100MB
maintenance_work_mem = 128MB
```

```
#max_fsm_pages = 204800  # min max_fsm_relations*16, 6 bytes each
max_fsm_pages = 2048000
#max_fsm_relations = 1000  # min 100, ~70 bytes each
max_fsm_relations = 10000
```

Note

If you increase memory settings for PostgreSQL, you will probably need to increase the maximum shared-memory settings in your OS. On Linux, you can do this by editing `/etc/sysctl` and adding the line: `kernel.shmmax=170639360`

Depending on how many shared memory segments you need, you may need to adjust that value.

5.2. Performance "Don't"s

Because of OpenNMS's high-I/O profile, there are a number of things that will cause performance issues on reasonably large installs.

- Don't run in a VM (although some pseudo-VMs like Xen are not as hard on I/O as things like VMware).
- Don't put the database or RRD data on file systems managed by LVM.
- Don't put DB or RRD data on file systems on RAID-5.
- Don't use older kernels. Linux 2.6 and Solaris 10 perform much better than older releases.

Chapter 6. Building From Source

6.1. Are you sure you want to do this?

OpenNMS is a complex software product, and it does not (yet) have a simple "`./configure && make && make install`" build process like many other tools. If there is a packaged release for your operating system, we highly suggest you use that instead. If you have problems with a packaged release, please see the troubleshooting section for assistance.

The best place to find out how to build OpenNMS is from the developer's page on the wiki. You will need to check out the code and then build it.

Chapter 7. Troubleshooting an OpenNMS Installation

7.1. Common Installation Issues

The following section contains advice for overcoming common installation issues. If your issue is not addressed below, please see the section on where to get help.

7.1.1. Dependency Problems

To assist with code management, the easiest way to install OpenNMS is via packages. Every effort has been made to insure that the packages OpenNMS depends on are required before the OpenNMS package can be installed. You should be able to find those packages on the distribution CDs that came with your system. For some of the more obscure packages, you can visit the OpenNMS FTP site and check in the `/pub/dependencies` directory. In addition, sites like Ibiblio and FreshRPMs are also good sources.

7.1.2. Error: "Started OpenNMS, but it has not finished starting up"

This can happen for a a number of reasons. You can run "`opennms -v status`" a few times after getting this error to see if OpenNMS eventually starts itself completely and if not, to see which daemons never start up completely. Here are some of the likely causes of this problem:

1. OpenNMS takes a while to startup. This can happen on larger installations and when this happens "`opennms -v status`" will eventually show that all services have started up. By default, the startup script will try 10 times to see if OpenNMS has started and will wait 5 seconds between each try. You can increase the number of times by creating `$OPENNMS_HOME/etc/opennms.conf` and adding a line like "`START_TIMEOUT=20`" to double the number of times it tests. You can set the value to `0` to have the startup script not wait for OpenNMS to start.

2. Database is not running. If only about half or less of the daemons are shown as running, you can check for this condition by looking for `FATAL` errors in the log files. You'll see something like "`Error accessing database`" in the logs.

3. Dhcpd doesn't start. See the item in the next section.

4. JNI library problem. OpenNMS uses a few native C libraries that are accessed using JNI (Java Native Interface). Normally they just work, except users have started seeing problems when running Linux in native AMD64 mode where they end up using a 32-bit (x86) version of Java and a 64-bit (AMD64) version of the JNI libraries, or vice-versa. If you have this problem, you might want to try switching your version of Java from 32-bit to 64-bit or in the other direction.

5. Other. If the OpenNMS is installed, and the packages were not forced in using options like "`--nodeps`", the application should run just fine. If not, OpenNMS has a robust logging facility. Change to the logs directory (usually `/var/log/opennms`) and search the logs, using `grep` or your tool of choice, for words like `FATAL` and `ERROR` (the two highest log severities). Those events should give you clues as to why OpenNMS is not working.

7.1.3. DHCP Poller Won't Start

The OpenNMS DHCP poller will fail to start most operating systems (Linux, in particular) if you are running a DHCP client on the OpenNMS server. You'll see this by running "`opennms -v status`" and seeing everything in the `running` state, except for `Dhcpd`. The solution is to edit `$OPENNMS_HOME/etc/service-configuration.xml` and comment-out the "`<service>...</service>`" stanza for `Dhcpd`. For example, this is what the section would look like after modification to disable `Dhcpd`:

```
<!-- Commented out since we have a DHCP client on this server
<service>
        <name>OpenNMS:Name=Dhcpd</name>
        <class-name>org.opennms.netmgt.dhcpd.jmx.Dhcpd</class-name>
        <invoke pass="1" method="start"/>
        <invoke at="status" pass="0" method="status"/>
        <invoke at="stop" pass="0" method="stop"/>
</service>
-->
```

We discourage the running of OpenNMS on a server that is a DHCP client, both because OpenNMS may not be able to monitor DHCP servers on the network, and it is important that the monitoring server have a static IP address for receiving traps and to be reliant on as few network services as possible.

7.1.4. Error: "runjava: Could not find an appropriate JRE"

The `runjava` program is used to locate a suitable JRE for OpenNMS at install time that will be used for the installer and also for running OpenNMS after installation. See the section earlier in this manual on installing Java for OpenNMS. If you installed Java in a location that `runjava` cannot find, you can use its "`-f`" option to specify the JRE you want OpenNMS to use.

7.1.5. Error: "The database server's error messages are not in English ..."

You either need to set "`lc_messages = 'C'`" in your postgresql.conf file and restart PostgreSQL or upgrade to PostgreSQL 7.4 or later.

The installer does not always verify that an operation will succeed before executing the operation (e.g.: dropping database functions). In this case, it catches the exceptions returned from the database and checks the exception to see if it is an "okay" exception that should be ignored (e.g.: if the database function does not exist when attempting to drop a function).

In PostgreSQL 7.4 and later, a new client/server protocol is used (version 3, to be specific) that provides specific error codes intended for programmatic evaluation and we use these error codes if the server provides them. However for PostgreSQL versions before 7.4, we require that the database server error language be in English (the 'C' locale) so that we can parse the text error messages. If you are not running PostgreSQL 7.4 or newer, the installer executes a bogus query against the database and checks for an expected result in English.

7.1.6. Error: "Column X in new table has NOT NULL constraint ..."

This is a warning that the installer might not update tables successfully. Make sure that your database is backed up, and run the installer again with the "-N" option to ignore this check.

As an attempt to ensure that the install will complete successfully, a check is done to see if there might be any rows with NULL columns that might be inserted into a column in an upgrade table with a NOT NULL constraint. This usually happens when a previous run of the installer failed, or might be due to modifications to the database schema or a really old version of the schema.

7.1.7. Error: "One or more backup tables from a previous install still exists"

When the installer runs to upgrade the OpenNMS database from a previous install, it often updates table schemas. When it does this, it copies the data in a table to a temporary table (e.g.: the contents of `node` are copied into `node_old_11033991291234`). The original table is deleted, the new version of the table is created, the data in the temporary table is translated into the new table, and finally the temporary table is deleted.

Unfortunately, the installer cannot check for all problems that might break translation, so sometimes the translation step fails. In this case, the installer "reverts" the table it was processing by dropping the new table and moving the temporary table into its place.

Reverting the table in case of a problem is all good and well, but sometimes even it does not work properly, especially with older versions of the Java installer. If this happens, the temporary table (the one with "_old_" in it) is left with all of the old data. Until OpenNMS 1.1.5, this problem would not be caught the next time you ran the installer. The installer would see that you did not have the `node` table, for example, and happily continue to create a new one for you. This is bad, especially since you probably still have data that you care about that is now in the "old" table.

If you get this error, you will want to get rid of the table(s) containing "_old_", however you want to first check if they contain data. For example, if you have a single table, `node_old_11033991291234`, no other `node_old_*` tables, and no `node` table, you can simply rename the table:

```
# psql -h localhost -U opennms opennms
Welcome to psql 7.4.6, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

opennms=# ALTER TABLE node_old_11033991291234 RENAME TO node;
```

You can use the "\d" command within `psql` to see what other tables exist in your database. You can use "`SELECT count(*) from table;`" (fill in the table name for "table") to get a count of rows in any table. If you have empty tables, you can just drop them. If you have multiple tables with data, you will have to decide which table of data you want to keep or merge them. This is left as a (not so simple) exercise for the reader.

7.1.8. Error: "Table X contains N rows (out of M) that violate new constraint Y"

Over time OpenNMS extends its database schema to improve functionality. This error can happen because of the way certain administrative functions in older versions of OpenNMS functioned or if the database was modified outside of OpenNMS (the latter is common for larger sites). Over time OpenNMS has introduced additional foreign key constraints on its database. These are used to ensure internal database consistency when data in two tables are tied together by a shared key. For example, each event can have a pointer to the node that it is related to; there is a foreign key constraint that requires that an event *must not* point at a node that does not exist.

Starting with 1.1.5, when we upgrade the database schema, we first check for rows that violate any new foreign key constraints that might be added. There are three options to to fix these errors:

1. Remove the offending rows. This is suggested if the number of rows that violate the constraint is small in comparison to the total number of rows in the affected table and if you don't need the data. Use "`$OPENNMS_HOME/bin/install -C <constraint> -X`" to delete the offending rows.

2. Mark the key in the offending rows to NULL. This is suggested if you need to keep the data around or are not yet sure about what to do with it. Use "`$OPENNMS_HOME/bin/install -C <constraint>`" to mark the key column to NULL in the offending rows.

3. Fix the key in the offending rows. This is for advanced users and requires a good amount of effort. This is left as an exercise for the reader.

7.1.9. Error: "- adding iplike database function... <snip> org.postgresql.util.PSQLException: ERROR: could not access file '<snip>/lib/iplike.so': Permission denied"

The PostgreSQL server cannot access the iplike.so file. This could be due to the file itself not having appropriate permissions for the user that PostgreSQL runs as and/or one or more of the parent directories of the iplike.so not having appropriate permissions.

This error is seen even when running the installer as root because it is not OpenNMS nor the installer that cannot access the iplike.so file, but the PostgreSQL database. The installer instructs the PostgreSQL database to load the iplike.so and the PostgreSQL database server usually runs as a non-root user, so it is subject to filesystem access control checks like any other normal user. This is commonly seen when people install OpenNMS into a home directory for root or another user and the permissions on that home directory do not allow users other than the owner of the directory access.

7.1.10. Error: "- adding iplike database function... <snip> org.postgresql.util.PSQLException: ERROR: could not load library ..."

The latter part of the error could be something like "`<path>/iplike.so: cannot open shared object file: No such file or directory`" or "`ld.so.1: postgres: fatal: <path>/iplike.so: wrong ELF class: ELFCLASS32`".

The PostgreSQL server cannot load the `iplike.so` file. This is almost always caused by the PostgreSQL server and the iplike.so file being compiled for different processor instruction sets. This is commonly seen when the PostgreSQL server is compiled to use a 64-bit instruction set but the OpenNMS `iplike.so` shared object is compiled for a 32-bit instruction set, although the opposite is possible, as well. You can use the "`file`" command on `iplike.so` and the `postmaster` binary with PostgreSQL to check their instruction sets.

The easiest solution is to see if there is a packaged version of OpenNMS compiled for the same instruction set (32- or 64-bit) as your PostgreSQL server. The next easiest method for most users is to switch the PostgreSQL server to match the instruction set that the `iplike.so` file was compiled for. For advanced users, you can compile OpenNMS yourself to fit the processor set that you need. See this post to the discuss list for some pointers.

7.1.11. Error: "Exception in thread "main" org.postgresql.util.PSQLException: ERROR: relation "pg_user" does not exist" when running installer.

This error means the database was not created properly. Since the installer script is supposed to create the database, one might assume it is a problem with OpenNMS, but instead it is an issue with the SELinux portions of Red Hat 4 (and CentOS 4). Basically, the postgres init_db command is not able to write to /dev/null, and it fails without a useful error message.

To get around this, run the following commands:

1. stop postgres
2. rm -rf /var/lib/pgsql/data
3. /usr/sbin/setenforce 0
4. start postgres

Note that step 2 will delete any changes you made to the postgresql configuration files and you'll need to redo them.

7.1.12. Error: java.io.FileNotFoundException: ... (Permission denied)

An exact example of this error is: "`java.io.FileNotFoundException: /opt/opennms/etc/users.xml (Permission denied)`".

If the above error happens when using admin functions through the web interface, such as managing users, notifications, and adding nodes, then the Tomcat web server is running as a non-root user but you haven't changed the permissions on the configuration files so the Tomcat user can access them. Go back and follow the instructions earlier in the install guide on setting up Tomcat to run as a non-root user.

7.2. Where to Get Help

OpenNMS is a community supported project. Please keep that in mind when seeking help on the program, as no one gets paid to work on the project (unless it is through a commercial support contract).

7.2.1. The Release Notes

Check the release notes for this release. They are in the Documentation section of the OpenNMS project page at SourceForge.

7.2.2. The OpenNMS Web Site

The main OpenNMS site is a Wiki. As a community project, there is a lot of good advice and information available there. In particular, we suggest checking the above-mentioned release notes, the FAQ entries on the wiki, the bug database and, of course, Google, before posting to a mailing list.

7.2.3. The OpenNMS Mailing Lists

OpenNMS maintains a number of active mailing lists on SourceForge:

opennms-announce	A low traffic, moderated mailing list for OpenNMS announcements. All posts to this list are duplicated on the *opennms-discuss* list.
opennms-cvs	This is a fairly high traffic list of all updates to the Subversion repositories on SourceForge. Moderated. Only SVN updates are posted here (no discussion).
opennms-devel	This list is for discussion of development of the OpenNMS codebase.

opennms-discuss	This is the main OpenNMS discuss list. It's pretty friendly, and reasonably high-volume. It tends to focus on configuration issues and general discussion of network management, but pretty much anything goes here. However, it is suggested that installation-related issues go to the *opennms-install* list instead.
opennms-install	This is a great list for new users to OpenNMS. The main focus is installation issues (cleared up by this great documentation, right?) but most "newbie" questions are welcome here.
opennms-maps	OpenNMS has a network map feature, which includes code for automatically determining relationships between hosts (Linkd). This is the appropriate list for discussion of maps and the underlying Linkd code.
opennms-windows	A discussion list for people running OpenNMS on Windows.
opennms-francais	A list for discussion of OpenNMS in French.
opennms-italia	A list for discussion of OpenNMS in Italian.
opennms-ug-tokyo	A list for discussion of OpenNMS in Japanese, as well as general discussion among the Tokyo OpenNMS Users Group.
opennms-ug-uk	A list for discussion of OpenNMS in UK English for those who don't speak American English (OK, just kidding). Actually, a discussion list for the UK OpenNMS Users Group. ;)

The OpenNMS mailing lists are also archived at gmane.org.

7.2.4. Commercial Support

If you are using OpenNMS in a production environment, or are considering it, you might be interested in commercial support. The OpenNMS Group maintains the OpenNMS project, and we also offer support, training, consulting services and custom development.